Comp Stat - Assignment 2

Akos Engelmann, Gergely Paradi, Fabian Gallyas, Ipek Cakin, Simon Jasansky

2023-03-21

1 Bootstrap with exhaustive enumeration

For the code implementation of the first Tasks (exhaustive enumeration and gray codes) please see our the file assingment_2.Rmd in our Github repository at https://github.com/SimonJasansky/comp_stat_A1/tree/main/assignment2

To do a complete enumeration bootstrap, the number of possible arrangements of the data is n^n. In our case, $n = 15$, which gives $15^{15} \approx 4.38 * 10^{17}$ calculations. However, all possible 15^{15} arrangements of the data also include arrangements with the same set of observations, but in a different order. These arrangements will still give the same correlation coefficient. Thus, the correlation coefficient can be calculated for each unique set of observations, and then weighted according to the number of possible arrangements of this set. In our implementation of the exhaustive enumeration, we used this weighting of the potential sets, where we calculated the weight for a set of observations with the multinomial probability, i.e. the R function dmultinom(indices, prob = rep(1,15)), as also implemented in Seiler (2016).

Doing this, the number of calculations is reduced from n^n to $\binom{2n-1}{2n-1}$, which in our case corresponds to a reduction from $15^{15} \approx 4.38 * 10^{17}$ calculations to only $\binom{29}{14} = 77558760 \approx 7.7 * 10^7$ calculations.

As this number of calculations is still not feasible in the amount of time we have for this assignment and on the machines available to us, we only performed the complete enumeration bootstrap on the first 5% of total possible observation sets, in order to being able to interpolate the total runtime when using all possible observation sets. Using this 5% of the observation sets ran for 0.943 hours, which would interpolate to 18.86 hours when using all the all possible observation sets. Note that we did not parallelize the algorithm.

2 Using Gray codes for compositions to speedup computations

Gray code, also known as reflected binary code, is a binary numeral system where two consecutive values differ by only one bit. This property makes it useful for exhaustive enumeration in digital circuits and other applications where sequential values need to be generated in a specific order without repeating any value or missing any value Diaconisk (1994). While there exists multiple literature about how to construct binary gray codes consisting only of 1s and 0, the literature on grey codes for combinations is limited. When creating gray codes for combinations, it is crucial that two consecutive codes differ only in two bits, and in each by only by the value of one, i.e. that only one observation is removed from the temporary sample, and another observation inserted. For our implementation fo the creation of grey codes we used the non-recursive algorithm provided by Diaconisk (1994), translating it from S to R.

Again, using this 5% of the observation sets, the program ran for 12.34 minutes, which would interpolate to 2.469 hours when using all the all possible observation sets. Note that this was also done without parallelization.

3 How much speedup can you get by using Gray codes? Show empirically or experimentally

The speedup achieved by using Gray codes depend on the size of the data set and the number of variables. Generally, the larger the data set and the higher the number of variables, the greater the speedup that can be achieved.

By using grey codes, only one observation gets exchanged with another, which allows to simply update the values and the correlation, instead of recalculating it over and over again. We obtained the formula for updating the values for calculating the Pearson correlation coefficient from Stackexchange (2018). Further, they are shown in Figures 1 and 2.

$$\hat{x} := \sum_{i=1}^{n} x_i \qquad \hat{y} := \sum_{i=1}^{n} y_i$$

$$\hat{a} := \sum_{i=1}^{n} x_i^2 \qquad \hat{b} := \sum_{i=1}^{n} y_i^2 \qquad \hat{c} := \sum_{i=1}^{n} x_i y_i$$

Figure 1: variables that are consecutively updated by subtracting the value of the deleted observation, and adding the value of the inserted observation.

$$r = \frac{\hat{c} - \frac{\hat{x}\hat{y}}{n}}{\sqrt{\hat{a} - \frac{\hat{x}^2}{n}}\sqrt{\hat{b} - \frac{\hat{y}^2}{n}}}$$

Figure 2: Formula used to calculate the Pearson correlation coefficient by updating.

As mentioned above, running the complete enumeration procedure on all possible observation sets would, by interpolation, take 18.86 hours. In comparison, using gray codes and the more efficient updating calculation of the correlation results in a total runtime of only 2.469 hours, equaling a reduction in runtime of -86.90 percent.

4 Which observation(s) do you need to remove from the sample to make the Monte Carlo and complete enumeration bootstrap look more similar?

To make the two more similiar, we could get rid of the outliers, namely 1 and possibly also 11, as determined by visual inspection of the scatter plot included in the homework assignment document.

Especially the first observation caused problems, as we usually tested the algorithms on the first 5% of possible observation sets, the first observation was typically heaviliy overrepresented (first obervation set was 15X first observation, second observation set was 14x first observation and 1x second observation, and so on). This often resulted in a negative correlation coefficient for multiple observation sets. Figure 3 shows the histogram of the correlation coefficients after running the gray code algorithm on the first 5% of observation sets, compared to the monte carlo simulation. It is very clear that the negative correlation and correlation around 0 are over-represented, compared to the Monte Carlo simulation.

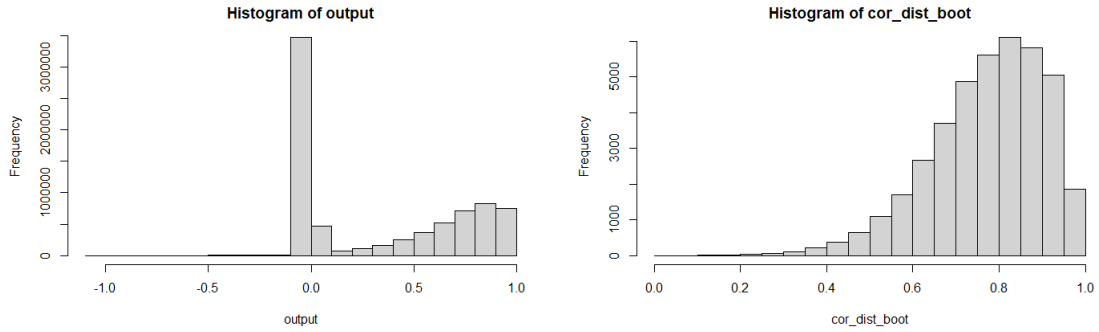

Figure 3: Histogram of the first 5 percent of observation sets using gray code algorithm (left), histogram of monte carlo simulation (right)

With the same reasoning, it could be argued that also observation 7 and 12 should be removed, although to a weaker extent. Diaconisk (1994) also shows in his paper that removing the first observation removes the "bump" around 0.8 and lets the correlation coefficient peak at a bit above 0.9.

5 Explain why you obtain difference results for Monte Carlo and complete enumeration bootstrap.

Monte Carlo bootstrap generates bootstrap samples by resampling from the original data set with replacement, while complete enumeration bootstrap generates all possible bootstrap samples. The differences between Monte Carlo and complete enumeration bootstrap results occurs, because Monte Carlo samples are a random subset of all possible samples, however complete enumeration samples include all possible samples. To compare these methods; bootstrapping uses the original, initial sample as the population from which to resample, but, on the other hand, Monte Carlo simulation is based on setting up a data generation process. Therefore, Monte Carlo bootstrap is subject to sampling variability, on the other hand complete enumeration bootstrap is not. As a result, Monte Carlo bootstrap tends to produce more variable results than complete enumeration bootstrap.

Furthermore, since with MLE we take the mean of the sample, any outlier can distort the mean and thus the MLE estimate. When we build our bootstrap, the repeated "biased" mean gets used. With the complete enumeration this is not the case, since we use every instance of our sample.

6 References

Diaconisk, Persi. 1994. "Gray Codes for Randomization Procedures." *Statistics and Computing* 21 (75): 287–302. https://statweb.stanford.edu/~cgates/PERSI/papers/graycodes.pdf.

Seiler, Cristof. 2016. "The Bootstrap (Part 2)." https://christofseiler.github.io/stats205/Lecture5/Bootstrap Part2.html#1.

Stackexchange. 2018. "Online Update of Pearson Coefficient." https://statweb.stanford.edu/~cgates/PERSI/ papers/graycodes.pdf.